Noname manuscript No.
(will be inserted by the editor)

Numerical simulation of skin transport using Parareal

Andreas Kreienbuehl · Arne Naegel · Daniel Ruprecht · Robert Speck · Gabriel Wittum · Rolf Krause

Received: date / Accepted: date

Abstract *In-silico* investigation of skin permeation is an important but also computationally demanding problem. To resolve all scales involved in full detail will not only require exascale computing capacities but also suitable parallel algorithms. This article investigates the applicability of the time-parallel Parareal algorithm to a brick and mortar setup, a precursory problem to skin permeation. The C++ library `Lib4PrM` implementing Parareal is combined with the `UG4` simulation framework, which provides the spatial discretization and parallelization. The combination's performance is studied with respect to convergence and speedup. It is confirmed that anisotropies in the domain and jumps in diffusion coefficients only have a minor impact on Parareal's convergence. The influence of load imbalances in time due to differences in number of iterations required by the spatial solver as well as spatio-temporal weak scaling is discussed.

Keywords Skin transport · Parareal · space-time parallelism · weak scaling · load balancing

This research is funded by the Deutsche Forschungsgemeinschaft (DFG) as part of the "Exasolvers" project in the Priority Programme 1648 "Software for Exascale Computing" (SPPEXA) and by the Swiss National Science Foundation (SNSF) under the lead agency agreement as grant SNF-145271. The research of A.K., D.R., and R.K. is also funded through the FUtuRe SwIss Electrical InfraStructure (FURIES) project of the Swiss Competence Centers for Energy Research (SCCER).

A. Kreienbuehl · D. Ruprecht · R. Krause
Institute of Computational Science, Faculty of Informatics, Università della Svizzera italiana, Via Giuseppe Buffi 13, CH-6904 Lugano, Switzerland

Arne Naegel · Gabriel Wittum
Goethe-Center for Scientific Computing, Goethe-University Frankfurt, Kettenhofweg 139, 60325 Frankfurt a.M., Germany

Robert Speck
Forschungszentrum Jülich GmbH, Institute for Advanced Simulation, Jülich Supercomputing Centre, Wilhelm-Johnen-Strasse, DE-52425 Jülich, Germany

1 Introduction

Permeation of chemical substances through human skin is an interesting and important process e.g. for the development of cosmetics or drugs. *In-vitro* studies with humans constitute the "gold standard" but they are expensive and limited by ethical and practical concerns. Here, *in-silico* studies are a viable alternative. They have been successfully used in the past (cf. the reviews in [1,24]) and can be expected to become even more important in the future. They allow for hypothesis testing and may lead to experiments through which effects not known today could be discovered.

Yet, numerical simulations in this field are demanding in terms of computational resources. The problem covers vastly different physical scales and, in case a complex full-fledged model is used, massive computational parallelism needs to be exploited to reach reasonable times-to-solutions. Therefore, many interesting aspects such as substructures of lipid bilayers or networks of keratin filaments [36,37] have not yet been investigated in three spatial dimensions (3D) using numerical simulations. Finally, modern imaging techniques make resolving a spectral range of a few nanometers possible, which results in "big data" for analyses. Understanding the functional mechanism of the skin is thus a candidate from the life sciences for applying exascale computing.

Considering the technology trend towards more and more parallelism, the application of new parallel methods to the problem investigated here becomes relevant.

Promising candidates for such methods are parallel-in-time integration methods that can add a direction of concurrency in addition to spatial parallelization, e.g., as used here, parallel multi-grid. In recent years, time-parallel methods have matured from a mainly mathematical concept to an approach with demonstrated efficiency in massively parallel computations [33,30]. They have been listed as a direction for mathematical research with the potential to help reaching exascale computing [11].

One of the most widely investigated parallel-in-time methods is Parareal, introduced in 2001 by Lions, Maday and Turinici [19]. It has been used for benchmark problems motivated by applications from fields as diverse as plasma physics [32], computational fluid dynamics [8,25] or quantum chemistry [7]. Improvements with respect to implementation are considered e.g. in [5,12]. Parareal's most appreciated aspect is probably that it is non-intrusive and rather easy to integrate into existing codes. Its drawback, on the other hand, is a quite severe bound on achievable parallel efficiency. However, because several other "across-the-step" time-parallel methods share similar features with Parareal (e.g. PITA [15], MGRIT [14] or PFASST [13]), studying Parareal's performance often already gives important insights.

Theoretical estimates for stability and convergence of Parareal for linear diffusive problems with constant coefficients can be found in [16]. Theory for diffusive problems with constant coefficients can also be found in [6]. For 2D diffusion with space-time dependent coefficients, numerical experiments showed only a marginal reduction in convergence speed [31]. In [4], the small impact of a time dependent viscosity on Parareal's convergence for an advection-diffusion problem is demonstrated. *However, performance for a 3D diffusive problem on a complex geometry with anisotropies has not yet been studied.*

In preparation for the eventual application of Parareal to skin permeation, this article provides an investigation of Parareal's performance for a 3D brick and mortar problem. From our point of view, this model serves as an excellent benchmark because it features challenges resulting from a complex anisotropic geometry and from jumping coefficients, which need to be resolved adaptively over long time intervals. Although locally the mathematical formulation of the brick and mortar problem is clear, the global picture is highly complex and linked to a real world application requiring a sound simulation infrastructure in terms of numerical methods and software. Here, we employ the simulation framework `UG4` [35] for the spatial discretization and linear solvers, for which excellent parallel scaling has been demonstrated [26]. We parallelize `UG4`'s serial temporal solvers through the C++ Parareal library `Lib4PrM`, which is integrated as a plugin.

The present article establishes the principle applicability of Parareal to the 3D brick and mortar problem. In doing so, it identifies a set of relevant issues to be tackled in order to develop an improved parallel-in-time integrator that can deliver reasonable efficiency for the skin transport problem. In particular, load balancing in time is identified as a critical issue when combining implicit methods for a complex PDE with Parareal. Because the number of iterations of the spatial solver varies in time, balancing temporal subintervals in Parareal simply by the number of time steps induces load imbalances which can affect speedup.

2 Problem and methods

As a test case, we study a simplified version of the 3D brick and mortar problem introduced earlier in [27,23]. The benchmark is defined on a biphasic domain $\Omega \subset \mathbb{R}^3$ that consists of two disjoint subsets $\Omega_{\mathrm{cor}}, \Omega_{\mathrm{lip}} \subset \mathbb{R}^3$ representing the so called corneocyte and lipid phase respectively. To be more specific, Ω is the interior of the union $\overline{\Omega}_{\mathrm{cor}} \cup \overline{\Omega}_{\mathrm{lip}}$ of closures. On this geometry we solve a diffusion equation with space-dependent diffusion coefficients. The evolution of the drug concentration $c_{\mathrm{p}}(\mathbf{x},t)$, $\mathrm{p} \in \{\mathrm{cor,lip}\}$, is modeled by the equation

$$\partial_t c_{\mathrm{p}}(\mathbf{x},t) = \nabla \cdot (D_{\mathrm{p}}(\mathbf{x}) \nabla c_{\mathrm{p}}(\mathbf{x},t)) \tag{1}$$

with $\mathbf{x} \in \Omega_{\mathrm{p}}$, $t \in [0,T]$, and a phase-dependent diffusion coefficient

$$D_{\mathrm{p}}(\mathbf{x}) = \begin{cases} D_{\mathrm{cor}} : \mathbf{x} \in \Omega_{\mathrm{cor}}, \\ D_{\mathrm{lip}} : \mathbf{x} \in \Omega_{\mathrm{lip}}. \end{cases} \tag{2}$$

For the simulation time we use $T = \frac{\lambda^2}{6D_{\mathrm{eff}}}$ which is the characteristic *lag time* of the problem. It is defined in terms of the membrane thickness λ and the effective (homogenized) diffusion coefficient D_{eff} [23]. In terms of boundary conditions we have an interior phase boundary $\Gamma = \overline{\Omega}_{\mathrm{cor}} \cap \overline{\Omega}_{\mathrm{lip}}$ and an exterior boundary $\partial\Omega$. For the exterior boundary we consider a mix of Dirichlet and homogeneous Neumann conditions, i.e. $\partial\Omega = \partial\Omega_{\mathrm{D}} \cup \partial\Omega_{\mathrm{N}}$ with

$$c_{\mathrm{p}}(\mathbf{x},t)|_{\partial\Omega_{\mathrm{D}}} = g(\mathbf{x}), \quad \mathbf{n} \cdot \nabla c_{\mathrm{p}}(\mathbf{x},t)|_{\partial\Omega_{\mathrm{N}}} = 0, \tag{3}$$

where $\mathbf{n}$ denotes the outward pointing normal vector on $\partial\Omega$. At the phase boundaries Γ, the flux must be continuous, i.e.

$$D_{\mathrm{lip}} \nabla c_{\mathrm{lip}}(\mathbf{x},t) \cdot \mathbf{n} = D_{\mathrm{cor}} \nabla c_{\mathrm{cor}}(\mathbf{x},t) \cdot \mathbf{n}. \tag{4}$$

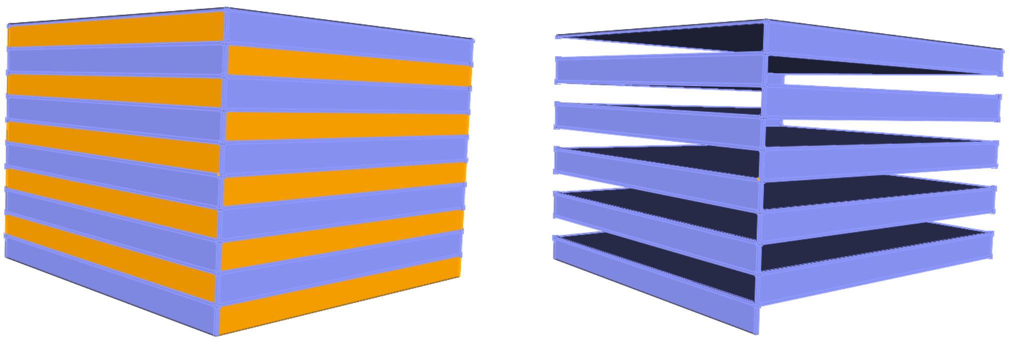

(a) with cornoecytes (yellow) (b) corneocytes removed

Fig. 1 A sketch of the 3D brick and mortar problem is shown. The geometry has ten layers of corneocytes (Ω_{cor}, yellow) that are embedded in a matrix of lipid bilayers (Ω_{lip}, blue).

Along Γ, concentrations can be discontinuous. However, they are often assumed to be linked by a Nernst's equilibrium $K_{\mathrm{cor/lip}} c_{\mathrm{lip}} = c_{\mathrm{cor}}$. When $K_{\mathrm{cor/lip}}$ is constant, the model may be reformulated, e.g., in terms of a continuous concentration [27,23]. Hence this work employs the simplifying assumption $c_{\mathrm{lip}} = c_{\mathrm{cor}}$. Note that more complex situations with locally fluctuating or even concentration dependent coefficients also play an important role [3,2,28].

The geometry used in this article is a 3D brick and mortar configuration as depicted in Figure 1. It features ten layers of corneocytes (yellow) that are embedded in a matrix of lipid bilayers (blue). It is a simplified version of more elaborate tetrakaidekahedral models with hybrid grids presented previously [23]; the brick and mortar model only consist of hexahedra with a reduced level of anisotropy. However, since jumping coefficients are present, it features some of the issues one encounters also in more complex situations.

Figure 2 shows the computed solution at three different times. To allow for a view into the interior of the domain, a cuboid representing a quarter of the total domain has been removed in the representation. Initially (not shown), the solution is zero on the whole domain with a Dirichlet boundary condition of $c_{\mathrm{p}}(\mathbf{x}, t) = 1$ on the top side. Then, the tracer starts to diffuse downwards in the lipid channels and much slower in comparison through the corneocytes. In the first subfigure, diffusion has just started and filled the upper half of the domain but the concentration is still essentially zero in the lower half. In the last subfigure, at the end of the simulation, the tracer has diffused down through the whole domain. Concentrations continue to change with smaller changes from step to step, eventually approaching a steady state. A prospective 3D model with a fully resolved lipid-bilayer substructure, as suggested for two dimensions in [36,37], will feature a similar effect on an even smaller scale in time and space.

2.1 Parareal

Let the temporal domain $[0, T]$ be decomposed into N_{t} subintervals $[t_j, t_{j+1}]$, $j = 0, \dots, N_{\mathrm{t}} - 1$, such that $t_0 = 0$, $t_{N_{\mathrm{t}}-1} = T$ and

$$[0, t_1] \cup \ldots \cup [t_{N_{\mathrm{t}}-1}, T] = [0, T]. \tag{5}$$

Let $\mathcal{C}$ and $\mathcal{F}$ be a "coarse" and "fine" time integration method[1] with time step size Δt and $\delta t \ll \Delta t$, respectively. For the sake of simplicity assume that all subintervals have the same length and that a constant number of both δt and Δt steps cover a subinterval exactly. Then, instead of serially integrating across $[0, T]$ with $\mathcal{F}$, Parareal uses the iteration

$$c_{n+1}^{k+1} = \mathcal{C}(c_n^{k+1}) - \mathcal{C}(c_n^k) + \mathcal{F}(c_n^k) \tag{6}$$

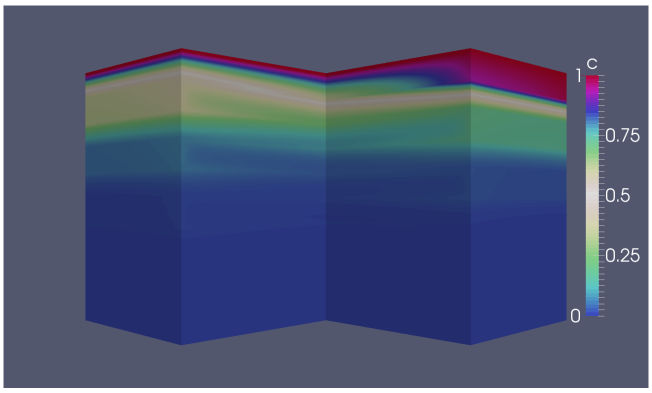
with k as iteration index. For the first subinterval, i.e. for $[0, t_1]$, set

$$c_0^k = c_0 \tag{7}$$

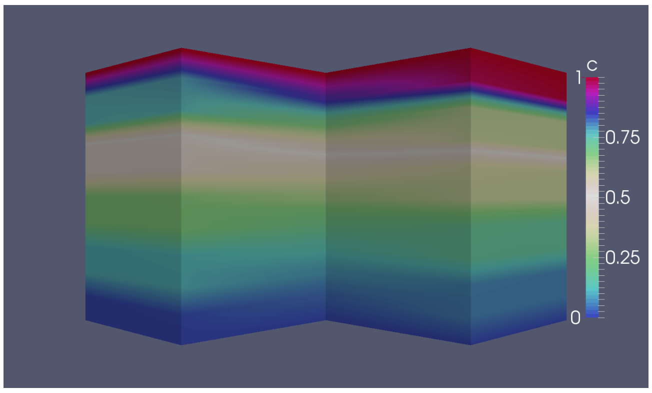
for all k, where c_0 is the given initial value. Note that as the iteration converges and $c_{n+1}^{k+1} - c_{n+1}^k$ approaches zero for all $n = 0, \dots, N_{\mathrm{t}} - 1$, the Parareal solution c_{n+1}^{k+1} converges to the serial fine solution $\mathcal{F}(c_n)$.

Formulation (6) introduces concurrency because as soon as the iterates c_n^k are known, the computationally expensive evaluation of the fine method can be done in parallel over all subintervals. That is, the time spent using the fine method in parallel equals the runtime of the fine method across a single subinterval instead of

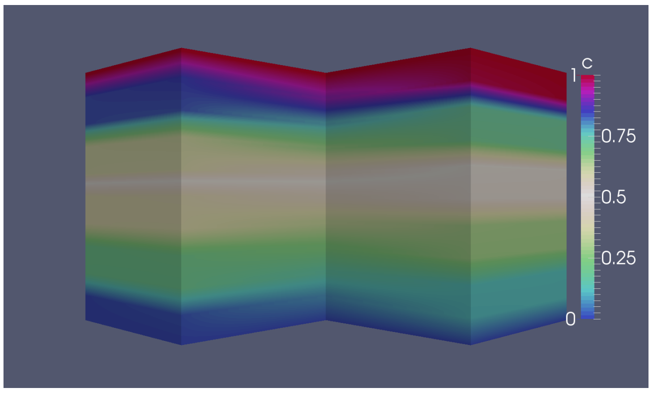
[1] The coarse method is often represented by $\mathcal{G}$, probably because of the French word "gros" for coarse.



(a) Time $t = T/16$.



(b) Time $t = T/2$.



(c) Time $t = T$.

Fig. 2 Solution at $t = T/16$ (top), $t = T/2$ (middle), and $t = T$ (bottom) where T denotes the lag-time of the problem. The block in the front has been removed to allow for a view into the interior of the computational domain.

the full interval $[0, T]$. However, multiple iterations are typically required and the propagation of corrections by the coarse method remains serial in time. Speedup therefore depends on finding a cheap enough coarse integrator that still leads to convergence in a small number of iterations.

2.2 Speedup from Parareal

Denote by N_c the number of coarse time steps per subinterval, by N_f the number of fine time steps per subinterval and by N_t the number of subintervals, which is assumed to be equal to the number of processors in the temporal direction. Further, denote by N_i the number of Parareal iterations and by τ^c and τ^f the computational runtime for a coarse or fine time step, respectively. If one assumes that every time step takes the same amount of time, speedup from Parareal can be modeled by

$$S(N_\mathrm{t}) \leq \frac{1}{\left(1 + \frac{N_\mathrm{i}}{N_\mathrm{t}}\right) \frac{N_\mathrm{c}}{N_\mathrm{f}} \frac{\tau^\mathrm{c}}{\tau^\mathrm{f}} + \frac{N_\mathrm{i}}{N_\mathrm{t}}}. \tag{8}$$

See e.g. [20] or [4] for a more detailed discussion of this bound.

For larger end times for the brick and mortar setup, however, this bound is too optimistic, because as the solution approaches a steady-state, the spatial solver requires fewer and fewer iterations per time step, making later time steps cheaper. In the numerical simulations presented below, we use a final time T for which the solution is still sufficiently far away from the steady-state and this effect is minimal, but we also discuss a formula valid for non-constant runtimes per time step. Here, to simplify the notation, we omit the index range for sums, maxima and minima; it is always implied to be $n = 0, \dots, N_\mathrm{t} - 1$. Now, denote by γ_n^c and γ_n^f the cost of running the coarse and fine method across the subinterval $[t_n, t_{n+1}]$. Then, a serial run of the fine or coarse method amounts to the duration

$$\Gamma_\mathrm{f} = \sum \gamma_n^\mathrm{f}, \qquad \Gamma_\mathrm{c} = \sum \gamma_n^\mathrm{c} \tag{9}$$

while a Parareal run with N_i iterations costs

$$\Gamma_\mathrm{P} = \Gamma_\mathrm{c} + N_\mathrm{i} \gamma^\mathrm{x}, \qquad \gamma^\mathrm{x} \equiv \max_n \left\{ \gamma_n^\mathrm{c} + \gamma_n^\mathrm{f} \right\}, \tag{10}$$

as the runtime for the parallel fine solve will be dominated by the subinterval with the longest simulation time. Also, using proper pipelining, the parallel runtime of the serial coarse correction step will be governed by the most expensive subinterval for $\mathcal{C}$. The resulting estimate for the speedup of Parareal is then

$$S(N_\mathrm{t}) = \frac{\Gamma_\mathrm{f}}{\Gamma_\mathrm{P}} = \frac{1}{\frac{\Gamma_\mathrm{c}}{\Gamma_\mathrm{f}} + N_\mathrm{i} \frac{\gamma^\mathrm{x}}{\Gamma_\mathrm{f}}} \tag{11}$$

This is a slight generalization of (8) in the sense that if $\gamma_n^\mathrm{c} = N_\mathrm{c} \tau^\mathrm{c}$ and $\gamma_n^\mathrm{f} = N_\mathrm{f} \tau^\mathrm{f}$ are constant for all subintervals, we get

$$\Gamma_\mathrm{c} = N_\mathrm{t} N_\mathrm{c} \tau^\mathrm{c}, \qquad \Gamma_\mathrm{f} = N_\mathrm{t} N_\mathrm{f} \tau^\mathrm{f}, \tag{12}$$

and

$$\gamma^{\mathrm{x}} = N_{\mathrm{c}}\tau^{\mathrm{c}} + N_f\tau^{\mathrm{f}} \tag{13}$$

for which (11) simplifies to (8). According to (11), in the case of imbalances in the distribution of computational load across subintervals, possible speedup is limited by the subinterval with the longest runtime for both the fine and coarse method.

The optimal configuration therefore corresponds to equal computing times for all subintervals. For explicit schemes, where the cost per time step is more or less constant, this balance is relatively easy to achieve by making sure every subinterval handles the same number of coarse and fine steps, resulting in the simple speedup model (8). For implicit methods, however, cost per time step is mainly determined by the cost of the spatial solver (typically depending mainly on the number of iterations), which in turn depends on the *a priori* unknown dynamics of the solution. Therefore, naively load balancing Parareal with implicit methods based on the number of time steps alone can lead to a significant loss in efficiency. Unfortunately, it seems that devising a proper load balancing in time for the implicit case is not straightforward and, to the best of our knowledge, has not yet been addressed in the literature. The easiest approach may be to use the information from the initial coarse run in Parareal to determine the size of the subintervals but this requires a non-negligible amount of implementation, might inhibit proper pipelining when requiring synchronization at some point and is thus left for future work.

To illustrate the effect of load imbalances in time on speedup from Parareal, Figure 3 visualizes both the projected speedup from (8) and (11): the ideal case assumes a constant coarse-to-fine ratio of

$$\frac{N_{\mathrm{c}}}{N_{\mathrm{f}}} = \frac{\gamma_n^{\mathrm{c}}}{\gamma_n^{\mathrm{f}}} = \frac{1}{10}. \tag{14}$$

The imbalanced case artificially increases $\gamma_3^{\mathrm{f}} = (1+b) \times 10$ and reduces $\gamma_2^{\mathrm{f}} = (1-b) \times 10$ while keeping all other γ_n^{f} and all γ_n^{c} unchanged. Here, b is an artificial parameter modeling load imbalance between the second and third slice in the formula for projected speedup. For $b = 0$, both slices have the same load ("ideal case") while increasing b corresponds to an increasing imbalance in load: for $b = 1$, the second slice no longer does any work while the third slice does twice as much work as in the ideal case. Note that the sum Γ_{f}, that is the total workload, remains constant. Clearly, the introduced imbalance has a noticeable detrimental effect on the projected speedup.

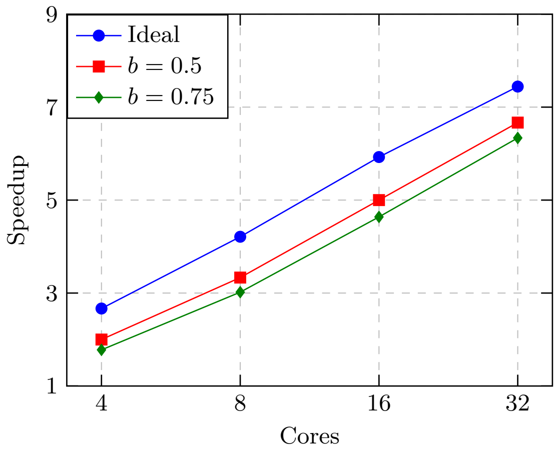


Fig. 3 Projected speedup for the ideally load balanced case and a case where the second and third slice are imbalanced by a factor of $b = 0.5$ or $b = 0.75$.

2.3 Weak scaling

When doubling the spatial resolution, the resulting increase in the computational cost per time step can be compensated for by a corresponding increase in the number of cores used for the spatial parallelization – at least if both the employed method and implementation show good weak scaling. For the spatial solver and parallelization of `UG4` applied to the benchmark used here, this has been demonstrated successfully in [34]. However, when the number of fine time steps N_{f} is also doubled, twice as many time steps have to be computed in order to keep the ratio $\delta t/\delta x$ constant, which leads to a doubling of time-to-solution (see also the discussion in [4]). Time parallelization can provide some mitigation by also doubling the number of subintervals and thus of cores used to parallelize along time - unless this leads to a massive increase in the number of required iterations.

However, because of the serial coarse correction, just as Parareal cannot achieve ideal strong scaling, it can also not provide 100% efficiency in weak scaling. To see this, let h denote a parameter governing accuracy of the discretization. Estimated runtime for a Parareal run on N_{t} cores (in time) is then

$$R_{2h}(N_{\mathrm{t}}) = N_{\mathrm{t}}N_{\mathrm{c}}\tau_{2h}^{\mathrm{c}} + N_{\mathrm{i}}\left(N_{\mathrm{c}}\tau_{2h}^{\mathrm{c}} + N_{\mathrm{f}}\tau_{2h}^{\mathrm{f}}\right). \tag{15}$$

Doubling spatial resolution, i.e. using h instead of $2h$ and performing twice as many fine and coarse steps on twice as many subintervals (but keeping the number of coarse and fine steps per subinterval constant) gives

$$R_h\left(2N_{\mathrm{t}}\right) = 2N_{\mathrm{t}}N_{\mathrm{c}}\tau_h^{\mathrm{c}} + N_{\mathrm{i}}\left(N_{\mathrm{c}}\tau_h^{\mathrm{c}} + N_{\mathrm{f}}\tau_h^{\mathrm{f}}\right) \tag{16}$$

assuming the number of iterations does not change. In the case of perfect weak scaling in space, by increasing the number of spatial cores the runtime per step can be kept constant, that is $\tau^{\mathrm{c}}_{2h} = \tau^{\mathrm{c}}_{h}$ and $\tau^{\mathrm{f}}_{2h} = \tau^{\mathrm{f}}_{h}$. Under this assumption, the projected weak scaling efficiency of the space-time parallelization is

$$\frac{R_{2h}(N_{\mathrm{t}})}{R_h(2N_{\mathrm{t}})} = \frac{N_{\mathrm{t}}\sigma + N_{\mathrm{i}}\,(1+\sigma)}{2N_{\mathrm{t}}\sigma + N_{\mathrm{i}}\,(1+\sigma)} < 1 \tag{17}$$

with $\sigma := N_{\mathrm{c}}\tau^{\mathrm{c}}_{h}/N_{\mathrm{f}}\tau^{\mathrm{f}}_{h} > 0$. Therefore, while weak scaling can never be perfect, a cheap enough coarse integrator ($\sigma \ll 1$) should still allow for good weak scaling – as long as the underlying spatial solver shows good weak scaling and the number of iterations is not affected by the increasing number of subintervals.

2.4 Implementation

For Parareal we use the C++ library `Lib4PrM` that uses MPI for the necessary communication of volume data in time. A straightforward approach to implementing (6) is sketched as pseudo code e.g. in [4]. Here, however, we use a somewhat more elaborate implementation that is based on the following observation. After the first iteration on the first subinterval $[0, t_1]$, the coarse terms in the Parareal iteration (6) cancel out, resulting in

$$c_1^{k+1} = \mathcal{F}\,(c_0) \tag{18}$$

for $k \geq 1$. That is, after one iteration, the first subinterval is guaranteed to have converged and the processors responsible for the first subinterval can "retire". After the second iteration, by the same argument, this will then be true for the processors handling the second subinterval and so on. After k iterations, the time-parallel fine method is guaranteed to have converged on the first k subintervals and all processors with an MPI rank (in time) smaller or equal to k could in principle be used otherwise. Put differently, Parareal converges always at least at a rate of one subinterval per iteration and when $k = n$ the Parareal method is guaranteed to have converged at $t_k \leq t_n$. While leaving processors idle according to this implementation does not affect runtime negatively, it has the potential to reduce the energy cost of a simulation, particularly in combination with "dynamic voltage and frequency scaling" [10]. This will be studied in a future work [17]. Also, if not enough processors are available to cover the whole interval $[0, T]$ by subintervals of a given size, converged processors could pick up subintervals at the end in a caterpillar-like way. Such even more involved implementations are left for future studies, though. Finally, in production runs one could also use some tolerance e.g. for the updates between iterations or a proper residual to decide when the solution at the end of a subinterval is converged [29].

2.5 Spatio-temporal discretization and solvers

Discretization in space and time is provided by the software package `UG4` [35]. We employ a plain vanilla vertex centered finite volume scheme in space that is combined with an implicit Euler scheme in time. For each time step this gives rise to a large linear system of equations, where the number of degrees of freedom corresponds to the number of vertices of the grid. The solver is a multigrid method with three steps of damped ($\omega = 0.6$) Jacobi relaxation used for pre- and post-smoothing. More details are provided in [34]. The coarse grid problem with 7‘581 degrees of freedom was solved using sequential SuperLU [18,9].

3 Numerical results

We report results from solving the brick and mortar problem described in §2 with the simulation framework described above. For both $\mathcal{C}$ and $\mathcal{F}$ we use an implicit Euler method with the time step size Δt for $\mathcal{C}$ being significantly larger than the time step size δt for $\mathcal{F}$.

All runs are performed on the Cray XC40 Piz Dora supercomputer at the Swiss National Supercomputing Centre (CSCS) in Lugano, Switzerland. This supercomputer is equipped with 1‘256 compute nodes, each of which consists of two 12-core Intel Xeon E5-2690v3 CPUs, making for a total of 30‘144 compute cores.[2] Its peak performance is 1.254 PFlops, placing it at position 56 in the Top500 November, 2014 list.[3] As compiler we used version 4.9.2 of the the GNU compiler collection[4] and, in the following, report runtimes of simulations without I/O operations.

3.1 Convergence of Parareal

Generally speaking, convergence of Parareal is affected by a number of parameters: The time step sizes and methods used for $\mathcal{C}$ and $\mathcal{F}$, the number of concurrently computed subintervals, the discretization used for the spatial derivatives, and the dynamics of the problem to be solved. To measure convergence, below the relative defect d_n^k between Parareal after k iterations and the fine solution run in serial is reported, i.e.

$$d_n^k = \frac{\left\|c_n^k - c_n\right\|_2}{\left\|c_n\right\|_2},\ n = 0, \ldots, N_{\mathrm{t}} - 1 \tag{19}$$

with

$$c_n = \mathcal{F}(c_{n-1}),\ n = 1, \ldots, N_{\mathrm{t}} - 1. \tag{20}$$

[2] `http://user.cscs.ch/computing_systems/piz_dora/`
[3] `http://www.top500.org/list/2014/11`
[4] `https://gcc.gnu.org`

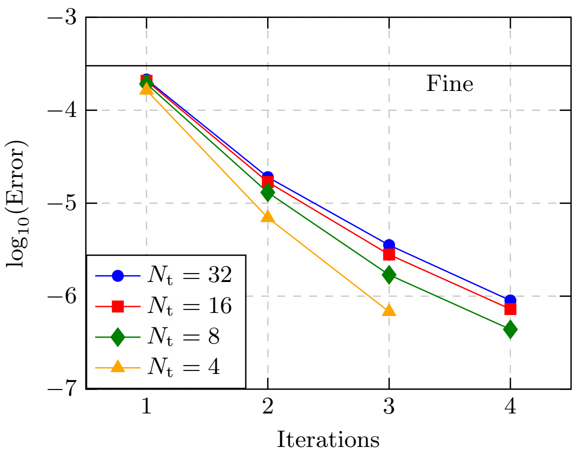


Fig. 4 The defect $d^k_{N_t-1}$ between the Parareal and the serial fine solution at $t = T$ versus the number of Parareal iterations for different numbers of subintervals N_t is illustrated. The discretization error of the serial fine method is indicated by the black horizontal line.

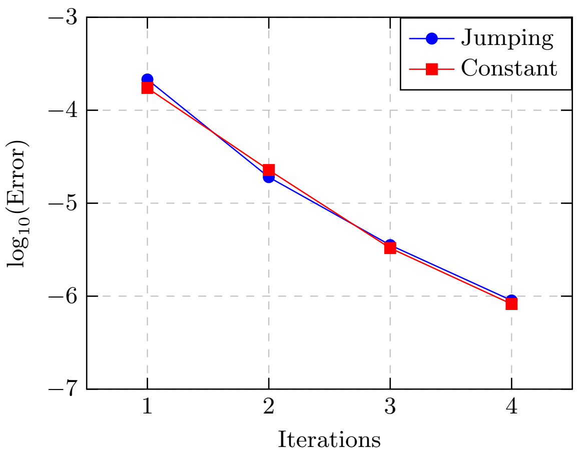


Fig. 5 The defect $d^k_{N_{t-1}}$ at $t = T$ between the Parareal $N_t = 32$ solution and the fine serial solution versus the number of Parareal iterations for the brick and mortar problem (red) and a reference configuration with constant diffusion coefficients (blue) is shown.

In order to avoid distortions through I/O times, only the final solution values are written out and the defect $d^k_{N_t-1}$ is reported except for in section §3.3. There, the defect is analyzed not only as a function of the number of iterations but also time.

Figure 4 shows the defect $d^k_{N_t-1}$ versus the number of iterations k. In addition, the estimated discretization error of the fine method resulting from a comparison against a run of $\mathcal{F}$ with a time step four times smaller than δt is shown. Parareal converges rapidly in all configurations. As can be expected, computing more subintervals in parallel slows down convergence somewhat. Here, for all $N_t \in \{4, 8, 16, 32\}$, one iteration suffices to reduce the defect below the discretization error of the fine method. *This confirms the usability of Parareal also for complex diffusion problems with anisotropic geometries and large jumps in the coefficients.* Particularly the relatively mild reduction of convergence speed as N_t is increased illustrates the potential for using larger numbers of cores to parallelize in time for this kind of problem, should a sufficiently large machine be available.

3.2 Effect of spatially varying coefficients

In the brick and mortar problem, the diffusion coefficients jump between $D_{\text{lip}} = 1$ in the lipid channels and $D_{\text{cor}} = 10^{-3}$ in the corneocytes. To assess the impact these jumps have on the convergence of Parareal, Figure 5 gives a comparison of the defect for the brick and mortar problem (red) and a reference configuration with $D_{\text{lip}} = D_{\text{cor}} = 10^{-3}$ throughout the whole domain. For the setup studied here, in line with the findings for 2D problems in [31], the jump in coefficients has almost no effect on how Parareal convergence. Experiments not documented here suggest that a larger T (that is, a final configuration closer to the steady state) can lead to a larger detrimental effect of coefficient jumps: However, even there this only resulted in a small number of additional iterations required for convergence.

3.3 Error over time

So far only the defect at the end of the simulation has been reported. In contrast, Figure 6 shows both the defect d^k_n of Parareal for $k = 1$ (red) and $k = 2$ (green) as well as the estimated discretization error of the coarse (blue) and fine (yellow) integrator. The figure shows the defect after every second subinterval for Parareal using $N_t = 32$.

Already after one iteration, the solution at $T = 1$ provided by Parareal has the same quality as when running the fine method serially. Therefore, speedup is reported below using $k = 1$. However, one iteration is not sufficient to reduce the defect of the whole transient to the discretization error: here, two iterations would be required after which the green line (Parareal) is completely below the yellow line (fine integrator).

3.4 Scaling of Parareal

Figure 7 shows the speedup from Parareal compared to running $\mathcal{F}$ in serial with the number of iterations chosen such that the defect of Parareal at $T = 1$ is below the estimated discretization error of $\mathcal{F}$ (for $N_t \in \{4, 8, 16, 32\}$

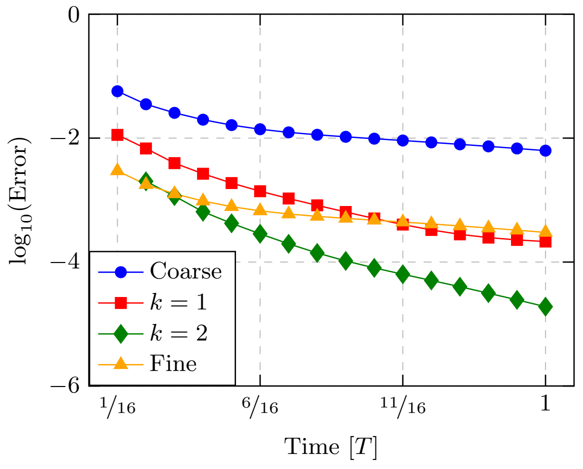


Fig. 6 The discretization error over time for the fine and coarse solution, and the defect of Parareal with $N_{\mathrm{t}} = 32$ subintervals after $k = 1$ and $k = 2$ iterations is illustrated.

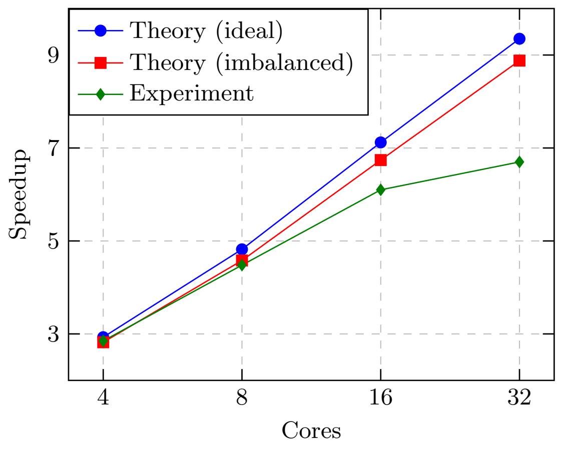


Fig. 7 Depicted is the speedup of the time parallelization with Parareal.

this means one iteration). The projected speedups for ideal load balancing according to (8) are marked by blue circles while the projected speedups according to (11), including differences in runtime between subintervals, are shown as red squares. Here, runtimes per subinterval γ_n^{c} and γ_n^{f} are measured experimentally from serial runs of $\mathcal{C}$ and $\mathcal{F}$. Measured speedups are shown as green diamonds. Up to sixteen subintervals, speedup follows the projected value reasonably well, but for 32 subintervals noticeable drop-off is observed – in small parts, this is due to imperfect load balancing as indicated by the difference between the red and blue curve. The major part, however, is overhead from communication and other factors, which are not incorporated in the speedup model.

3.5 Spatio-temporal weak scaling

Figure 8 shows convergence of Parareal runs for four different setups with increasing spatial and temporal resolution but keeping both the number of elements in space and time steps per core constant. The first one (blue) uses a time step size of $\Delta t = 1/8$ and $\delta t = 1/128$ in units of T. The second one (red) on the other hand uses half the coarse and fine time step and half the spatial mesh width so that $\delta t/\delta x$ and $\Delta t/\delta x$ are the same in both runs. It also uses twice as many cores in time and eight times more cores in space, so that both the number of elements per core and the number of time steps per core remain constant, too. The green and yellow line then correspond to runs that again double spatial and temporal resolution. Higher spatio-temporal resolution leads to smaller defects for Parareal, while the rates of convergence (i.e. the slopes of the lines) remain roughly the same.

Exact configurations are shown in Table 1: note how $\Delta t^{-1}/N_{\mathrm{t}}$ and $\delta t^{-1}/N_{\mathrm{t}}$ as well as # Elements/P_{space} are constant in all configurations. Also, each refinement step halves the estimated fine discretization error, which matches the behavior expected for the employed first-order discretization. The number of Parareal iterations required for convergence to the accuracy of $\mathcal{F}$ stays constant: every configuration is converged after one iteration. Runtimes are increasing as the problem size grows, so space-time weak scaling is not optimal. Partly, this is because of the overhead from the coarse method, see the discussion above, partly because of less than optimal weak scaling of the spatial solver. Nevertheless, Parareal helps to mitigate some of the increase in runtime from increasing the spatio-temporal problem size.

4 Conclusions

Computational modeling of skin permeation is of interest for different applications. However, the full problem requires resolution of a vast range of scales, leading to enormous computational requirements. Massively parallel computers are needed but these require suitable parallel numerical methods to be used efficiently.

This article investigates the applicability and performance of the time-parallel Parareal integrator to a relevant precursory problem of skin transport, namely a 3D brick and mortar configuration. For this, the C++ Parareal library `Lib4PrM` is integrated as a plug-in into the simulation environment `UG4` using implicit integrators and a geometric multi-grid as spatial solver. While the brick and mortar problem does not yet feature the same geometric level of detail as the skin transport

Run	Δt^{-1}	δt^{-1}	# Elements	P_{space}	N_{t}	P_{total}	e_{fine}	$d^1_{N_{\text{t}}}$	R [s]	Factor
$(2_{\text{t}}, 3_{\text{s}})$	8	128	277‘440	3	2	6	$10^{-2.9}$	$10^{-2.8}$	132.79	–
$(4_{\text{t}}, 24_{\text{s}})$	16	256	2‘219‘520	24	4	96	$10^{-3.2}$	$10^{-3.2}$	239.34	1.80
$(8_{\text{t}}, 192_{\text{s}})$	32	512	17‘756‘160	192	8	1‘536	$10^{-3.5}$	$10^{-3.7}$	316.83	1.32
$(16_{\text{t}}, 1‘536_{\text{s}})$	64	1‘024	142‘049‘280	1‘536	16	24‘576	$10^{-3.8}$	$10^{-4.3}$	508.70	1.61

Table 1 Configuration of the runs shown in Figure 8. Both the number of coarse and fine time steps per core in time and the number of elements per core in space are kept constant in all runs. Here, e_{fine} indicates the estimated discretization error of the fine integrator and $d^1_{N_{\text{t}}}$ is the defect after one iteration. R indicates runtime in seconds. Note that N_{t} is the number of subintervals and equal to the number of cores in time. The last column gives the factor between runtimes: ideal space-time weak scaling corresponds to a factor of 1.0, ideal spatial weak scaling with no time parallelization corresponds to a factor of two while no weak scaling at all would lead to a factor of $8 \times 2 = 16$ because the simulations use a 3D spatial discretization.

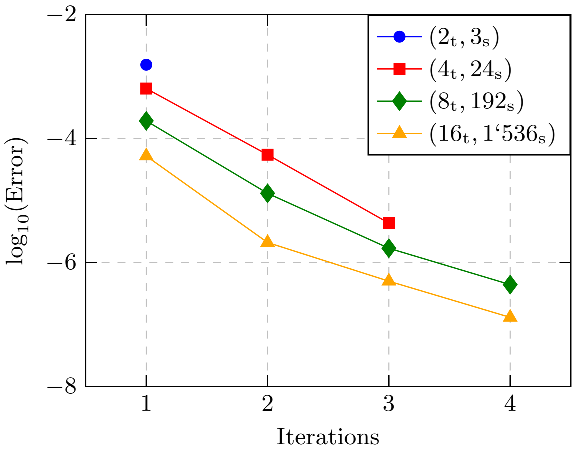


Fig. 8 The convergence of Parareal for four different configurations is shown. The ratios $\delta t/\delta x$ and $\Delta t/\delta x$ are kept constant and the number of degrees-of-freedom per core as well as the number of time steps per subinterval are kept constant, too. Thus, e.g. in the run with 8 cores in time and 192 cores in space the spatio-temporal resolution is twice as good as in the run with 4 cores in time and 24 cores in space. Note that because the problem is in 3D, doubling the spatial resolution requires eight times more cores.

problem, it already has jumps in the diffusion coefficients of several orders of magnitude on a highly anisotropic domain. The article is an extension of a previous study of a 2D problem on a domain with a much simpler structure [31].

Performance of the space-time parallel solver is studied in several numerical experiments. It is confirmed that Parareal still converges quickly for the brick and mortar case. Moreover, strong and weak scaling as well as implications for the “trap of weak scaling” are illustrated and discussed.

As the solution of the brick and mortar problem approaches a steady state in time, initial guesses for the geometric multi-grid become more accurate if time steps of constant length are used. This leads to faster convergence of the geometric multigrid, which in turn induces an imbalance in workload between the different processors in time. For the chosen setup, this effect is small but by deriving a simple theoretical model, imbalances in time are shown to have a potentially significant effect on parallel efficiency. For Parareal with implicit integrators applied to complex PDEs, the resulting load imbalance is an important issue that has to be addressed.

A number of possible directions for future research emerge from the experiments presented here. So far, coarsening in Parareal was done only in time by using a larger time step. Better results can be expected if the spatial discretization is coarsened simultaneously. This requires a closer integration of Parareal with the spatial multi-grid solver, in order to provide interpolation and restriction routines. Also, this approach can be taken further by interweaving iterations of the time-parallel method with iterations of the spatial solver, as discussed e.g. for Parareal in [22] or for PFASST in [21]. Another important issue that is also connected to load balancing is spatial and temporal adaptivity. While both can in principle be used in Parareal, they greatly complicate the load balancing problem. Finally, as energy consumption is becoming a more and more important issue in high-performance computing, a thorough benchmarking in terms of energy-to-solution is also an important direction for future work.

References

1. Advanced Drug Delivery Reviews: Modeling the human skin barrier - Towards a better understanding of dermal absorption **65**(2) (2013). DOI 10.1016/j.addr.2012.12.002
2. Anissimov, Y., Roberts, M.S.: Diffusion modelling of percutaneous absorption kinetics: 4. Effects of Slow Equilibration Process Within Stratum Corneum on Absorbtion and Desorption Kinetics. J Pharm Science **98**, 772–781 (2009). DOI 10.1002/jps.21461
3. Anissimov, Y.G., Roberts, M.S.: Diffusion modeling of percutaneous absorption kinetics: 3. Variable diffusion and partition coefficients, consequences for stratum corneum depth profiles and desorption kinetics. J Pharm Sci **93**(2), 470–487 (2004). DOI 10.1002/jps.10567. URL `http://dx.doi.org/10.1002/jps.10567`

4. Arteaga, A., Ruprecht, D., Krause, R.: A stencil-based implementation of Parareal in the C++ domain specific embedded language STELLA. Applied Mathematics and Computation (2015). URL `http://dx.doi.org/10.1016/j.amc.2014.12.055`
5. Aubanel, E.: Scheduling of Tasks in the Parareal Algorithm. Parallel Computing **37**, 172182 (2011). URL `http://dx.doi.org/10.1016/j.parco.2010.10.004`
6. Bal, G.: On the convergence and the stability of the parareal algorithm to solve partial differential equations. In: R. Kornhuber, et al. (eds.) Domain Decomposition Methods in Science and Engineering, *Lecture Notes in Computational Science and Engineering*, vol. 40, p. 426432. Springer, Berlin (2005). URL `http://dx.doi.org/10.1007/3-540-26825-1_43`
7. Bylaska, E.J., Weare, J.Q., Weare, J.H.: Extending molecular simulation time scales: Parallel in time integrations for high-level quantum chemistry and complex force representations. The Journal of Chemical Physics **139**(7), 074,114 (2013). URL `http://dx.doi.org/10.1063/1.4818328`
8. Celledoni, E., Kvamsdal, T.: Parallelization in time for thermo-viscoplastic problems in extrusion of aluminium. International Journal for Numerical Methods in Engineering **79**(5), 576598 (2009). URL `http://dx.doi.org/10.1002/nme.2585`
9. Demmel, J.W., Eisenstat, S.C., Gilbert, J.R., Li, X.S., Liu, J.W.H.: A supernodal approach to sparse partial pivoting. SIAM J. Matrix Analysis and Applications **20**(3), 720–755 (1999)
10. Dick, B., Vogel, A., Khabi, D., Rupp, M., Küster, U., Wittum, G.: Utilization of empirically determined energy-optimal cpu-frequencies in a numerical simulation code. Comp Vis Sci (accepted for publication)
11. Dongarra, J., al.: Applied Mathematics Research for Exascale Computing. Tech. Rep. LLNL-TR-651000, Lawrence Livermore National Laboratory (2014). URL `http://science.energy.gov/~/media/ascr/pdf/research/am/docs/EMWGreport.pdf`
12. Elwasif, W.R., Foley, S.S., Bernholdt, D.E., Berry, L.A., Samaddar, D., Newman, D.E., Snchez, R.S.: A dependency-driven formulation of parareal: parallel-in-time solution of PDEs as a many-task application. In: Proceedings of the 2011 ACM international workshop on many task computing on grids and supercomputers, p. 1524 (2011). URL `http://dx.doi.org/10.1145/2132876.2132883`
13. Emmett, M., Minion, M.L.: Toward an Efficient Parallel in Time Method for Partial Differential Equations. Communications in Applied Mathematics and Computational Science **7**, 105132 (2012). URL `http://dx.doi.org/10.2140/camcos.2012.7.105`
14. Falgout, R.D., Friedhoff, S., Kolev, T.V., MacLachlan, S.P., Schroder, J.B.: Parallel time integration with multigrid. SIAM Journal on Scientific Computing **36**, C635C661 (2014)
15. Farhat, C., Chandesris, M.: Time-decomposed parallel time-integrators: theory and feasibility studies for fluid, structure, and fluid-structure applications. International Journal for Numerical Methods in Engineering **58**(9), 13971434 (2003). URL `http://dx.doi.org/10.1002/nme.860`
16. Gander, M.J., Vandewalle, S.: On the Superlinear and Linear Convergence of the Parareal Algorithm. In: O.B. Widlund, D.E. Keyes (eds.) Domain Decomposition Methods in Science and Engineering, *Lecture Notes in Computational Science and Engineering*, vol. 55, p. 291298. Springer Berlin Heidelberg (2007). URL `http://dx.doi.org/10.1007/978-3-540-34469-8_34`
17. Kreienbuehl, A., Benedusi, P., Ruprecht, D., Krause, R.: Time parallel gravitational collapse simulation (2015). In preparation.
18. Li, X., Demmel, J., Gilbert, J., iL. Grigori, Shao, M., Yamazaki, I.: SuperLU Users' Guide. Tech. Rep. LBNL-44289, Lawrence Berkeley National Laboratory (1999). `http://crd.lbl.gov/~xiaoye/SuperLU/`. Last update: August 2011
19. Lions, J.L., Maday, Y., Turinici, G.: A "parareal" in time discretization of PDE's. Comptes Rendus de l'Acadmie des Sciences - Series I - Mathematics **332**, 661668 (2001). URL `http://dx.doi.org/10.1016/S0764-4442(00)01793-6`
20. Minion, M.L.: A Hybrid Parareal Spectral Deferred Corrections Method. Communications in Applied Mathematics and Computational Science **5**(2), 265301 (2010). URL `http://dx.doi.org/10.2140/camcos.2010.5.265`
21. Minion, M.L., Speck, R., Bolten, M., Emmett, M., Ruprecht, D.: Interweaving PFASST and parallel multigrid. SIAM Journal on Scientific Computing (2015). URL `http://arxiv.org/abs/1407.6486`
22. Mula, O.: Some contributions towards the parallel simulation of time dependent neutron transport and the integration of observed data in real time. Ph.D. thesis, Université Pierre et Marie Curie - Paris VI (2014). URL `https://tel.archives-ouvertes.fr/tel-01081601`
23. Naegel, A., Heisig, M., Wittum, G.: A comparison of two- and three-dimensional models for the simulation of the permeability of human stratum corneum. Eur J Pharm Biopharm **72**(2), 332 – 338 (2009). DOI DOI:10.1016/j.ejpb.2008.11.009. URL `http://www.sciencedirect.com/science/article/B6T6C-4V1KMMP-1/2/b906a3a90140385ba35b48bed48fdef7`
24. Querleux, B. (ed.): Computational Biophysics of the Skin. Pan Stanford Publishing (2014)
25. Randles, A., Kaxiras, E.: Parallel in time approximation of the lattice Boltzmann method for laminar flows. Journal of Computational Physics **270**, 577586 (2014). URL `http://dx.doi.org/10.1016/j.jcp.2014.04.006`
26. Reiter, S., Vogel, A., Heppner, I., Rupp, M., Wittum, G.: A massively parallel geometric multigrid solver on hierarchically distributed grids. Computing and Visualization in Science **16**(4), 151–164 (2013). DOI 10.1007/s00791-014-0231-x. URL `http://dx.doi.org/10.1007/s00791-014-0231-x`
27. Rim, J.E., Pinsky, P.M., van Osdol, W.W.: Using the method of homogenization to calculate the effective diffusivity of the stratum corneum with permeable corneocytes. Journal of Biomechanics **41**(4), 788–796 (2008). DOI 10.1016/j.jbiomech.2007.11.011. URL `http://www.sciencedirect.com/science/article/B6T82-4RWHXFR-2/2/bfe8e93f74d145a105071a106d6d227c`
28. Rim, J.E., Pinsky, P.M., van Osdol, W.W.: Multiscale modeling framework of transdermal drug delivery. Annals of Biomedical Engineering **37**(6), 1217–1229 (2009)
29. Ruprecht, D.: Convergence of Parareal with spatial coarsening. PAMM **14**(1), 10311034 (2014). URL `http://dx.doi.org/10.1002/pamm.201410490`
30. Ruprecht, D., Speck, R., Emmett, M., Bolten, M., Krause, R.: Poster: Extreme-scale space-time parallelism. In: Proceedings of the 2013 Conference on High Performance Computing Networking, Storage and Analysis Companion, SC '13 Companion (2013). URL `http:`

//sc13.supercomputing.org/sites/default/files/PostersArchive/tech_posters/post148s2-file3.pdf

31. Ruprecht, D., Speck, R., Krause, R.: Parareal for diffusion problems with space- and time-dependent coefficients. In: Domain Decomposition Methods in Science and Engineering XXII, *Lecture Notes in Computational Science and Engineering*, vol. 104, pp. 3–10. Springer International Publishing Switzerland (2015). URL http://dx.doi.org/10.1007/978-3-319-18827-0_1
32. Samaddar, D., Newman, D.E., Snchez, R.S.: Parallelization in time of numerical simulations of fully-developed plasma turbulence using the parareal algorithm. Journal of Computational Physics **229**, 65586573 (2010). URL http://dx.doi.org/10.1016/j.jcp.2010.05.012
33. Speck, R., Ruprecht, D., Krause, R., Emmett, M., Minion, M.L., Winkel, M., Gibbon, P.: A massively space-time parallel N-body solver. In: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis, SC '12, p. 92:192:11. IEEE Computer Society Press, Los Alamitos, CA, USA (2012). URL http://dx.doi.org/10.1109/SC.2012.6
34. Vogel, A., Calotoiu, A., Strube, A., Reiter, S., Nägel, A., Wolf, F., Wittum, G.: Automated performance modeling applied to the software framework ug4. submitted (2015)
35. Vogel, A., Reiter, S., Rupp, M., Nel, A., Wittum, G.: Ug 4: A novel flexible software system for simulating pde based models on high performance computers. Computing and Visualization in Science **16**(4), 165–179 (2013). DOI 10.1007/s00791-014-0232-9. URL http://dx.doi.org/10.1007/s00791-014-0232-9
36. Wang, T.F., Kasting, G.B., Nitsche, J.M.: A multiphase microscopic diffusion model for stratum corneum permeability. I. Formulation, solution, and illustrative results for representative compounds. J. Pharm. Sci. **95**(3), 620–648 (2006). DOI 10.1002/jps.20509. URL http://dx.doi.org/10.1002/jps.20509
37. Wang, T.F., Kasting, G.B., Nitsche, J.M.: A multiphase microscopic diffusion model for stratum corneum permeability. II. Estimation of physicochemical parameters, and application to a large permeability database. J. Pharm. Sci. **96**(11), 3024–3051 (2007). DOI 10.1002/jps.20883. URL http://dx.doi.org/10.1002/jps.20883